FIG. 1

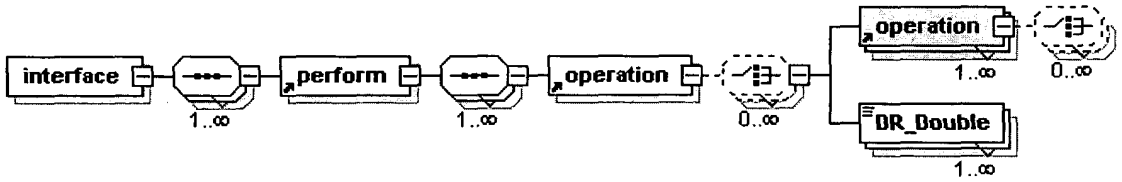

FIG. 2


```
<perform>
    <operation name="Sum">
        <operation name="GetComplexResult"/>
        <operation name="GetComplexResult"/>
        <operation name="GetComplexResult"/>
        <operation name="GetComplexResult"/>
                --- more here ---
        <operation name="GetComplexResult"/>
    </operation>
</perform>
```

FIG. 3


```
<perform>
    <operation name="Sum" parallel="yes">
        <operation name="GetComplexResult"/>
        <operation name="GetComplexResult"/>
        <operation name="GetComplexResult"/>
        <operation name="GetComplexResult"/>
                --- more here ---
        <operation name="GetComplexResult"/>
    </operation>
</perform>
```

FIG. 4


```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by Andrew Doddington (JP Morgan Chase) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault=
        "qualified"   attributeFormDefault="unqualified">
        <!-- FIrst define our core container - a request containing zero or more values or voids -->
        <xs:element name="request">
                <xs:annotation>
                        <xs:documentation>
                                A sequence of zero or more values and/or void results
                                        - including both inline constants and method calls.
                        </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:choice maxOccurs="unbounded">
                                <xs:element ref="AbstractValue"/>
                                <xs:element ref="AbstractVoid"/>
                        </xs:choice>
                        <xs:attribute name="service" type="xs:string" use="optional" default="edgService"/>
                </xs:complexType>
        </xs:element>
        <xs:element name="AbstractValue" abstract="true">
                <xs:annotation>
                        <xs:documentation>Represents an arbitrary value.</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="AbstractVoid" abstract="true">
                <xs:annotation>
                        <xs:documentation>
                                Represents the concept of something with *no* value.
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="DoubleValue" abstract="true" substitutionGroup="AbstractValue">
                <xs:annotation>
                        <xs:documentation>An arbitrary double precsion floating-point value.</xs:documentation>
                </xs:annotation>
        </xs:element>
        <xs:element name="DoubleConstant" type="xs:double" substitutionGroup="DoubleValue">
                <xs:annotation>
                        <xs:documentation>
                                A floating-point value, expressed as an in-line constant.
                        </xs:documentation>
                </xs:annotation>
        </xs:element>
```

FIG. 5


```
<xs:element name="FloatValue" abstract="true" substitutionGroup="DoubleValue">
	<xs:annotation>
		<xs:documentation>An arbitrary floating-point value.</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:element name="FloatConstant" type="xs:float" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>
			A floating-point value, expressed as an in-line constant.
		</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:element name="IntegerValue" abstract="true" substitutionGroup="AbstractValue">
	<xs:annotation>
		<xs:documentation>An arbitrary integer value.</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:element name="IntegerConstant" type="xs:integer" substitutionGroup="IntegerValue">
	<xs:annotation>
		<xs:documentation>A integer value, expressed as an in-line constant.</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:element name="StringValue" abstract="true" substitutionGroup="AbstractValue">
	<xs:annotation>
		<xs:documentation>An arbitrary string value.</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:element name="StringConstant" type="xs:string" substitutionGroup="StringValue">
	<xs:annotation>
		<xs:documentation>A string value, expressed as an in-line constant.</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:complexType name="FunctionDecl">
	<xs:annotation>
		<xs:documentation>The base type for all function blocks.</xs:documentation>
	</xs:annotation>
</xs:complexType>
<xs:complexType name="StructureDecl">
	<xs:annotation>
		<xs:documentation>The base type for all structs.</xs:documentation>
	</xs:annotation>
</xs:complexType>
```

FIG. 6


```
<xs:complexType name="SingleFloat">
	<xs:annotation>
		<xs:documentation>
			An argument list consisting of a single floating point value.
		</xs:documentation>
	</xs:annotation>
	<xs:complexContent>
		<xs:extension base="FunctionDecl">
			<xs:sequence>
				<xs:element ref="FloatValue">
					<xs:annotation>
						<xs:appinfo source=
							"http://www.jpmorgan.com/fort/procml/name">x</xs:appinfo>
					</xs:annotation>
				</xs:element>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
<xs:complexType name="ManyFloat">
	<xs:annotation>
		<xs:documentation>
			An argument list consisting of a single floating point value.
		</xs:documentation>
	</xs:annotation>
	<xs:complexContent>
		<xs:extension base="FunctionDecl">
			<xs:sequence maxOccurs="unbounded">
				<xs:element ref="FloatValue">
					<xs:annotation>
						<xs:appinfo source=
							"http://www.jpmorgan.com/fort/procml/name">X</xs:appinfo>
					</xs:annotation>
				</xs:element>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
```

FIG. 7


```
<xs:complexType name="Float_Float">
	<xs:annotation>
		<xs:documentation>An argument list consisting of a two floating point values.</xs:documentation>
	</xs:annotation>
	<xs:complexContent>
		<xs:extension base="FunctionDecl">
			<xs:sequence>
				<xs:element ref="FloatValue">
					<xs:annotation>
						<xs:appinfo source=
							"http://www.jpmorgan.com/fort/procml/name">X</xs:appinfo>
					</xs:annotation>
				</xs:element>
				<xs:element ref="FloatValue">
					<xs:annotation>
						<xs:appinfo source=
							"http://www.jpmorgan.com/fort/procml/name">Y</xs:appinfo>
					</xs:annotation>
				</xs:element>
			</xs:sequence>
		</xs:extension>
	</xs:complexContent>
</xs:complexType>
<xs:element name="Sum" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>
			A function that takes one or more floats and returns their sum.
		</xs:documentation>
	</xs:annotation>
	<xs:complexType>
		<xs:complexContent>
			<xs:extension base="ManyFloat"/>
		</xs:complexContent>
	</xs:complexType>
</xs:element>
<xs:element name="Product" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>
			A function that takes one or more floats and returns their product.
		</xs:documentation>
	</xs:annotation>
	<xs:complexType>
		<xs:complexContent>
			<xs:extension base="ManyFloat"/>
		</xs:complexContent>
	</xs:complexType>
</xs:element>
```

FIG. 8


```
<xs:element name="PercentageValue" abstract="true" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>A (logically) constrained floating-point value.</xs:documentation>
	</xs:annotation>
</xs:element>
<xs:element name="PercentageConstant" substitutionGroup="PercentageValue">
	<xs:annotation>
		<xs:documentation>Percentage constants can be physically constrained by the schema.</xs:documentation>
	</xs:annotation>
	<xs:simpleType>
		<xs:restriction base="xs:float">
			<xs:minInclusive value="0"/>
			<xs:maxInclusive value="100"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
<xs:element name="Person">
	<xs:annotation>
		<xs:documentation>A Person structure (with three fields:- Name/Age/Salary).</xs:documentation>
	</xs:annotation>
	<xs:complexType>
		<xs:complexContent>
			<xs:extension base="StructureDecl">
				<xs:sequence>
					<xs:element ref="StringValue">
						<xs:annotation>
							<xs:appinfo source=
                                "http://www.jpmorgan.com/fort/procml/name">name</xs:appinfo>
						</xs:annotation>
					</xs:element>
					<xs:element ref="IntegerValue">
						<xs:annotation>
							<xs:appinfo source=
                                "http://www.jpmorgan.com/fort/procml/name">age</xs:appinfo>
						</xs:annotation>
					</xs:element>
					<xs:element ref="FloatValue">
						<xs:annotation>
							<xs:appinfo source=
                                "http://www.jpmorgan.com/fort/procml/name">salary</xs:appinfo>
						</xs:annotation>
					</xs:element>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
</xs:element>
```

FIG. 9


```
<xs:element name="Negate" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>
			A function that takes a float and returns its negation.
		</xs:documentation>
	</xs:annotation>
	<xs:complexType>
		<xs:complexContent>
			<xs:extension base="SingleFloat"/>
		</xs:complexContent>
	</xs:complexType>
</xs:element>
<xs:element name="Add" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>
			A function that takes exactly two floating points values and returns their sum.
		</xs:documentation>
	</xs:annotation>
	<xs:complexType>
		<xs:complexContent>
			<xs:extension base="Float_Float"/>
		</xs:complexContent>
	</xs:complexType>
</xs:element>
<xs:element name="Multiply" substitutionGroup="FloatValue">
	<xs:annotation>
		<xs:documentation>
			A function that takes exactly two floating points values and returns their product.
		</xs:documentation>
	</xs:annotation>
	<xs:complexType>
		<xs:complexContent>
			<xs:extension base="Float_Float"/>
		</xs:complexContent>
	</xs:complexType>
</xs:element>
```

FIG. 10


```
<xs:element name="SavePersonDetails" substitutionGroup="AbstractVoid">
        <xs:annotation>
                <xs:documentation>Function that takes a person structure.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
                <xs:complexContent>
                        <xs:extension base="FunctionDecl">
                                <xs:sequence>
                                        <xs:element ref="Person">
                                                <xs:annotation>
                                                        <xs:appinfo source=
                                                           "http://www.jpmorgan.com/fort/procml/name">thePerson</xs:appinfo>
                                                </xs:annotation>
                                        </xs:element>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:element>
<xs:element name="ModifyPersonSalary" substitutionGroup="AbstractVoid">
        <xs:annotation>
                <xs:documentation>A function that takes a Person structure plus a float.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
                <xs:complexContent>
                        <xs:extension base="FunctionDecl">
                                <xs:sequence>
                                        <xs:element ref="Person">
                                                <xs:annotation>
                                                        <xs:appinfo source=
                                                           "http://www.jpmorgan.com/fort/procml/name">thePerson</xs:appinfo>
                                                </xs:annotation>
                                        </xs:element>
                                        <xs:element ref="FloatValue">
                                                <xs:annotation>
                                                        <xs:appinfo source=
                                                           "http://www.jpmorgan.com/fort/procml/name">newSalary</xs:appinfo>
                                                </xs:annotation>
                                        </xs:element>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
</xs:element>
</xs:schema>
```

FIG. 11


```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\development\procml\xsd\procml.xsd">
        <!-- A simple value -->
        <FloatConstant>3.14</FloatConstant>
        <PercentageConstant>3.14</PercentageConstant>
        <!-- A simple function call -->
        <Add>
                <!-- Argument 1 -->
                <FloatConstant>123.45</FloatConstant>
                <!-- Argument 2 -->
                <FloatConstant>567.89</FloatConstant>
        </Add>
        <!-- A simple nested function call -->
        <Negate>
                <Add>
                        <!-- Argument 1 -->
                        <FloatConstant>123.45</FloatConstant>
                        <!-- Argument 2 -->
                        <FloatConstant>567.89</FloatConstant>
                </Add>
        </Negate>
        <!-- A more complex nested function call -->
        <Multiply>
                <!-- Argument 1 -->
                <Add>
                        <!-- Argument 1.1 -->
                        <FloatConstant>123.45</FloatConstant>
                        <!-- Argument 1.2 -->
                        <FloatConstant>567.89</FloatConstant>
                </Add>
                <!-- Argument 2 -->
                <Negate>
                        <Add>
                                <!-- Argument 2.1 -->
                                <FloatConstant>111</FloatConstant>
                                <!-- Argument 2.2 -->
                                <FloatConstant>222</FloatConstant>
                        </Add>
                </Negate>
        </Multiply>
```

FIG. 12


```
<!-- A function call taking an arbitrary number of arguments (3 for this example) -->
<Sum>
        <!-- Argument 1 -->
        <Multiply>
                <!-- Argument 1.1 -->
                <Add>
                        <!-- Argument 1.1.1 -->
                        <FloatConstant>123.45</FloatConstant>
                        <!-- Argument 1.1.2 -->
                        <FloatConstant>567.89</FloatConstant>
                </Add>
                <!-- Argument 1.2  -->
                <Negate>
                        <!-- Argument 1.2.1 -->
                        <Add>
                                <!-- Argument 1.2.1.1 -->
                                <FloatConstant>111</FloatConstant>
                                <!-- Argument 1.2.1.2 -->
                                <FloatConstant>222</FloatConstant>
                        </Add>
                </Negate>
        </Multiply>
        <!-- Argument 2 -->
        <Add>
                <!-- Argument 2.1 -->
                <FloatConstant>123.45</FloatConstant>
                <!-- Argument 2.2 -->
                <FloatConstant>567.89</FloatConstant>
        </Add>
        <!-- Argument 3-->
        <Add>
                <!-- Argument 3.1 -->
                <FloatConstant>111</FloatConstant>
                <!-- Argument 3.2 -->
                <PercentageConstant>22</PercentageConstant>
        </Add>
</Sum>
```

FIG. 13


```
<!-- calling a void function that takes a Person structure -->
        <SavePersonDetails>
                <Person>
                        <StringConstant>John Doe</StringConstant>
                        <IntegerConstant>25</IntegerConstant>
                        <FloatConstant>10000.00</FloatConstant>
                </Person>
        </SavePersonDetails>
        <!-- calling a void function that takes a Person structure and a float value -->
        <ModifyPersonSalary>
                <Person>
                        <StringConstant>Jane Doe</StringConstant>
                        <IntegerConstant>35</IntegerConstant>
                        <FloatConstant>12000.00</FloatConstant>
                </Person>
                <FloatConstant>14000.00</FloatConstant>
        </ModifyPersonSalary>
</request>
```